Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Dean Koštomaj

Izkaznica NFC v postopku preverjanja študijskih obveznosti

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Mira Trebar

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Zamenjava klasičnih študentskih izkaznic z novimi karticami NFC (Near Field Communication) omogoča uporabo pametnih telefonov NFC tudi v študijskem procesu. Kandidat naj v diplomskem delu analizira tehnologijo NFC v povezavi z razvojem aplikacij za Android operacijski sistem in kartice NFC (TDS_DESF_EV1, ICODE SLI) ter postopke uporabe v študijskem procesu. Zasnuje in izdela naj mobilno aplikacijo, ki bo omogočala uvoz podatkov s seznamom udeležencev iz datoteke .csv ali preko spletne storitve iz študijskega sistema. Uporabniku naj omogoča enostaven in hiter postopek preverjanja prisotnosti študentov, vnos ocen ali točk, pregled podatkov ter izvoz podatkov v datoteko .csv za nadaljnjo obdelavo. Delovanje aplikacije naj preveri na izmišljenih testnih podatkih in opiše njene prednosti in nadaljnje možnosti za dejansko uporabo v prihodnosti, ko bodo kartice NFC v uporabi.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Dean Koštomaj, z vpisno številko **63110295**, sem avtor diplomskega dela z naslovom:

Izkaznica NFC v postopku preverjanja študijskih obveznosti

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Mire Trebar,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 14. septembra 2014 Podpis avtorja:

Zahvalil bi se vsem, ki so kadarkoli pripomogli pri izdelavi te diplomske naloge. Posebna zahvala gre mentorici doc. dr. Miri Trebar, za pomoč in usmerjanje pri izdelavi ter staršem, sorodnikom in moji punci za podporo.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
API	application program interface	programski vmesnik
CPU	central process unit	centralno procesna enota
CSV	character separated values	z znakom ločene vrednosti
IDE	integrated development environment	integrirano razvojno okolje
ISO	international organization for standardization	mednarodna organizacija za standardizacijo
JSON	JavaScript Object Notation	JavaScript objektna notacija
NDEF	NFC data exchange format	NFC format za izmenjavo podatkov
NFC	near field communication	komunikacija kratkega dosega
P2P	peer-to-peer	enak z enakim
RAM	random access memory	bralno-pisalni pomnilnik
RFID	radio frequency identification	radiofrekvenčna identifikacija
SDK	software development kit	nabor razvojnih orodij
SHA	secure hash algorithm	kriptografska zgoščevalna funkcija
SQL	simple query language	strukturirani povpraševalni jezik
UID	unique identifier	unikatni identifikator
URI	uniform resource identifier	enotni označevalnik vira

Povzetek

Za diplomsko nalogo smo si zastavili zanimivo nalogo, da bi razvili mobilno aplikacijo, ki bi profesorjem olajšala preverjanje prisotnih študentov na izpitnih rokih in vajah. Za ta namen je bilo potrebno raziskati tehnologijo NFC (Near Field Communication) v študentskih izkaznicah. Rešitev je aplikacija za NFC mobilne naprave, ki jih poganja operacijski sistem Android. Izkazalo se je, da so za uporabnika pomembne še druge funkcionalnosti, ki poenostavijo uvoz in izvoz seznamov ter vnašanje podatkov povezanih s prisotnostjo, kot sta točkovanje ali ocenjevanje. Vse funkcionalnosti programa so opisane, prav tako pa je predstavljeno njihovo delovanje, ki ima vključeno povezavo z informacijskim sistemom za pridobivanje seznamov študentov. Aplikacija je bila razvita s pomočjo Android SDK orodij, Java programskega jezika ter Eclipse razvojnega okolja. V postopku evalvacije se bo prototipna verzija uporabljala za preverjanje prisotnosti na Fakulteti za računalništvo in informatiko.

Ključne besede: NFC, Android, SQLite, izkaznica, prisotnost, preverjanje.

Abstract

For this thesis we had an interesting idea to develop a mobile application that will make checking the presence of students at exams and practises easier. First the NFC (Near Field Communication) in students identification cards had to be studied. The solution that was made is an application for mobile devices that runs on Android operating system. It turned out that the user needs also other functionalities, such as easy importing and exporting of lists, writing and saving students grades. All the functionalities are described and explained how they work including the connection to information system for importing lists. The application was developed with Android SDK development kit, Java programming language and Eclipse IDE. In the evaluation process, the prototype version will be used to verify the presence of students at the Faculty of Computer and Information Science.

Keywords: NFC, Android, SQLite, identification card, presence, verification.

Poglavje 1

Uvod

Živimo v obdobju, v katerem je čas vedno bolj pomemben. Skoraj vse korporacije, ki nam ponujajo najrazličnejše izdelke, od najmanjših programov do največjih strojev, nam poskušajo olajšati opravila in nam prihraniti nekaj dragocenega časa. Delo pa lahko olajšamo tudi profesorjem. Verjetno ga ni profesorja, ki mu ne bi ugajala popolna avtomatizacija preverjanja prisotnosti študentov na vajah in izpitih.

Velika večina profesorjev že ima pametne telefone, s katerimi lahko opravljajo najrazličnejše aktivnosti. Zakaj ne bi z njimi preverjali tudi prisotnosti študentov? Profesor bi si lahko prisotne študente vpisoval kar v telefon. Z novejšimi študentskimi izkaznicami obstaja še lažji način. Te izkaznice imajo že vgrajen NFC čip, na katerem so shranjeni podatki študenta. Seveda je potrebno imeti na napravi podprto NFC tehnologijo, če želimo odčitavati tovrstne študentske izkaznice. Danes tudi to ni več tako velika težava. Veliko novejših pametnih telefonov ima vgrajeno NFC napravo.

V diplomski nalogi je predstavljena mobilna aplikacija, ki rešuje zastavljen problem. S pomočjo NFC tehnologije odčitava študentske izkaznice in na ta način preverja prisotnost študentov. Pri razvoju je bil cilj ustvariti aplikacijo, ki je čim lažja za uporabo, saj je glavni cilj poenostavljanje opravil. Za čim boljšo izvedbo se je bilo potrebno seznaniti s tehnologijo NFC in njenim delovanjem na mobilnih napravah. V diplomski nalogi so predstavljene osnove te tehnologije in opisane vrste naprav, ki jo uporabljajo. Ker je rešitev implementirana za mobilne naprave z Android operacijskim sistemom, je predstavljeno tudi delovanje te tehnologije z

omenjenim operacijskim sistemom.

Poglavje 2

Teoretični del

2.1 Tehnologija NFC

Komunikacija kratkega dosega, ali angleško *Near Field Communication* (v nadaljevanju **NFC**), nam omogoča prenos podatkov na kratki razdalji. Gre za nadgradnjo tehnologije RFID (*Radio frequency identification*). Čeprav sta si obe tehnologiji na prvi pogled zelo podobni, je med njima kar nekaj razlik. Ena od najpomembnejših razlik je, da lahko z RFID tehnologijo samo beremo značke, na katerih je kaj zapisano, ali pa na prazne pišemo. Pri NFC tehnologiji, pa je dostikrat prejemnik podatkov neka naprava. Napravi, ki komunicirata preko NFC, lahko izvesta veliko druga o drugi. Na primer lahko izvesta, ali ima njena sogovornica še kakšno drugo tehnologijo za prenos podatkov (Bluetooth, Wi-Fi). Tako se lahko dogovorita, da bo nadaljnji prenos podatkov potekal preko ene od omenjenih tehnologij, ki sta veliko hitrejši [1].

2.1.1 Delitev NFC naprav

- **Pasivne NFC naprave**
  Prva skupina so pasivne naprave. Tu gre za značke, ki lahko upravljajo z manjšo količino podatkov in nimajo lastnega vira napajanja. Te naprave ne procesirajo podatkov, ki jih prejmejo in se ne morejo povezovati z drugimi pasivnimi napravami.
- **Aktivne NFC naprave**

Druga skupina pa so aktivne naprave. Tu gre za naprave, ki imajo lasten vir napajanja. Lahko se povezujejo tako z aktivnimi, kot tudi s pasivnimi NFC napravami. Najboljši primer aktivne NFC naprave je pametni telefon [2].

2.1.2 Kako NFC deluje?

Tako kot druge tehnologije za prenos podatkov tudi NFC deluje preko radijskih valov. Komunikacija kratkega dosega temelji na starejši RFID ideji, ki uporablja elektromagnetno indukcijo. Prav to nam omogoča, da lahko odčitavamo pasivne naprave, ki nimajo lastnega napajanja. Ob približanju pasivne naprave z aktivno, se ustvari elektromagnetno polje, kar omogoča napajanje in delovanje pasivne naprave.

Pošiljanje podatkov poteka na frekvenci 13,56 MHz, na razdalji do 10 cm. Doseg NFC tehnologije je namenoma tako kratek, saj to omogoča zelo nizko porabo energije. To pa je še posebej koristno pri napravah, ki jih napajajo baterije. Hitrost prenosa podatkov je 106, 212 ali 424 kilobitov na sekundo.

2.1.3 NDEF sporočila

Podatki, ki se prenašajo preko NFC tehnologije, so zapisani v formatu NDEF (NFC Data Exchange Format). To je ena od ključnih stvari, ki jo doda NFC RFID tehnologiji [1]. Vsako sporočilo formata NDEF (ang. *NDEF message*) je lahko sestavljeno iz enega ali več zapisov (ang. *NDEF record*), ki vsebuje tip zapisa, unikaten identifikator, dolžino podatkov in podatke [3].

Vsaka naprava z vgrajeno NFC tehnologijo naj bi prepoznala štiri osnovne tipe zapisov podatkov [1].

- **Zapis s preprostim besedilom (ang. *simple text record*)**

  Ta zapis vsebuje besedilo, ki ga želimo poslati. Kot meta podatke lahko vsebuje tudi jezik, v katerem je besedilo napisano in kodiranje znakov. Po navadi ne vsebuje nikakršnih navodil za ciljno napravo.

- **URI zapis (ang. *URI record*)**

Gre za zapis, ki vsebuje spletni naslov. Naprava ob prejetju takega zapisa po navadi odpre brskalnik in naloži spletno stran, ki je na poslanem naslovu.

- **Pameten plakat (ang. *smart poster*)**

  Zapis te oblike po navadi vsebuje dodatne informacije, ki jih pripnemo plakatu. Naprava lahko ob prejetju tega zapisa odpre katero od aplikacij.

- **Podpis (ang. *signature*)**

  Tovrstni zapis nam zagotavlja pošiljanje verodostojnih informacij o izvoru podatkov vsebovanih v NDEF zapisu.

2.1.4 Arhitektura NFC tehnologije

V grobem lahko NFC razdelimo na tri plasti. Prva plast je fizična plast [1]. V to plast spada CPU, protokoli, ki skrbijo za komunikacijo med napravama (to so: UART (ang. *universal asynchronous receive-transmit*), SPI (ang. *serial peripheral interface*), I2C (ang. *interintegrated circuit communication*) in USB (ang. *universal serial bus*)), radio kontrolerji in radio specifikacije. Za njih se uporablja standard ISO-14443-2. Nato imamo srednjo plast, ki je sestavljena iz dela za okvirjanje podatkov ter ukaznih protokolov. Okvirjanje podatkov poteka po standardu ISO-14443-3. Ukazni protokoli pa so razdeljeni v dve skupini. Za branje NFC značk in pisanje nanje se uporabljajo ukazni protokoli, ki so zgrajeni po standardu ISO-14443A. Za P2P (*peer-to-peer*) povezavo med napravama pa se uporabljajo protokoli standarda ISO-18092. Zadnja plast je aplikacijska plast, ki skrbi za format sporočila in vsebuje uporabniški vmesnik [1].

2.2 NFC in operacijski sistem Android

Kot že vemo, ima operacijski sistem Android podprto komunikacijo kratkega dosega. To ne pomeni, da ima vsaka Android naprava že vgrajeno to tehnologijo. Zato se bomo tukaj osredotočili na njih. Android naprave so sposobne branja NFC značk in komuniciranja z drugimi Android napravami preko te tehnologije. Omogočajo tudi tri glavne načine delovanja [4]:

- **bralno/pisalni način** (ang. reader/writer mode), ki omogoča branje in pisanje na NFC značke,
- **P2P način** (ang. P2P mode), nam omogoča prenašanje podatkov med dvema NFC napravama (to uporablja Android Beam) in
- **način posnemanja kartice** (ang. card emulation mode), ki nam omogoča, da se naša naprava obnaša kot NFC kartica. Tako lahko beremo podatke z nje z NFC čitalnikom.

Branje NFC značk z operacijskim sistemom Android

Ob približanju NFC značke operacijski sistem Android značko zazna in prebere NDEF podatke na njej s sistemom za odpremo značk (ang. *tag dispatch system*) (slika 2.1). Ta sistem analizira zaznano značko, kategorizira njene podatke in odpre aplikacijo, ali pa ponudi seznam teh, ki jih prebrana vsebina zanima [6].

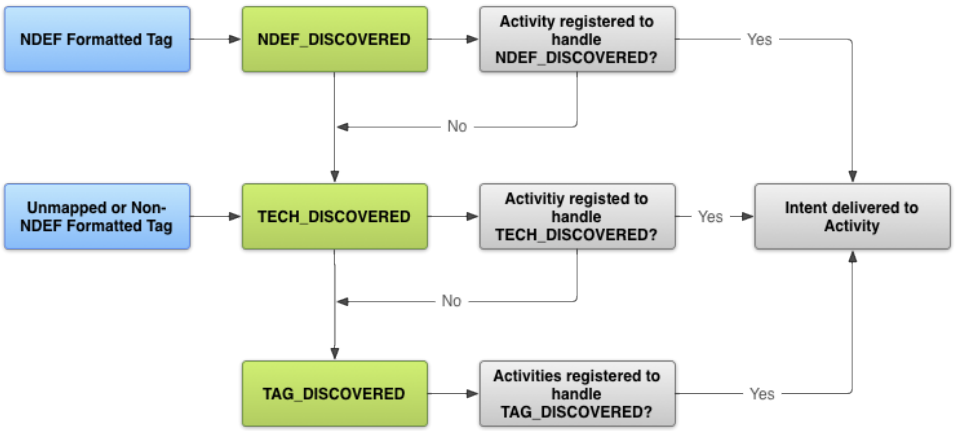
Sistem za odpremo značk po odčitavanju značke ustvari namen (ang. *intent*). Poznamo tri različne tipe namenov:

- **NDEF:** se ustvari, kadar je sporočilo formata NDEF. Ta namen ima najvišjo prioriteto.
- **TECH:** v primeru, da se nobena aplikacija ne odzove na NDEF namen, sistem za odpremo značk pregleda, ali se katera od aplikacij odzove na TECH namen. Ta namen se požene prvi, ne da bi se pognal NDEF namen, v primeru, da je odčitano sporočilo formata NDEF in ga ni mogoče povezati z nobenim URI ali MIME (ang. *multipurpose internet mail extension*) tipom.
- **TAG:** ta namen ima najnižjo prioriteto in se sproži v primeru, da se nobena aplikacija ne odzove na ostala dva namena.

V primeru, da se na nobenega od naštetih namenov ne odzove niti ena aplikacija, ta namen ne naredi ničesar. Kako deluje sistem za odpremo značk v operacijskem sistemu Android, je prikazano na sliki 2.1.

Sistem za prioritetno odpremo značk

Sistem omogoča, da naša aplikacija prevzame prioriteto nad drugimi za obdelavo namena, ki ga ustvari sistem za odpremo značk. Gre za sistem, ki ga je potrebno



Slika 2.1: Delovanje sistema za odpremo značk.

aktivirati, ko je naša aplikacija v ospredju in za pravilno delovanje tudi onemogočiti ob izhodu. Sistemu je potrebno določiti, za katere vrste značk želimo prevzeti prioriteto za obdelavo namena. Tako Android ne bo, ob približanju pravega tipa značke, čakal na odziv ostalih aplikacij, ampak bo namen prepustil kar naši. To pomeni, da tudi v primeru zainteresiranosti neke druge aplikacije za ta tip značke, nam Android ne bo ponudil možnosti, da jo odpremo, ker je že naša aplikacija v ospredju [7].

2.3 Druge tehnologije

2.3.1 Programska oprema

Eclipse IDE Juno

Eclipse je odprtokodno razvojno okolje. Projekt je bil ustvarjen s strani IBM podjetja v novembru 2001. Do konca leta 2003 je skupina razvijalcev tega okolja narastla že na 80 članov. 2. februarja leta 2004, pa je sledila reorganizacija v neprofitno korporacijo [9].

Android SDK

Android SDK je programski paket, ki omogoča razvoj aplikacij za Android platformo. Vsebuje primere projektov z izvorno kodo, razvojna orodja, Android Emulator in potrebne knjižnice za izdelavo Android aplikacij. Aplikacije so napisane v programskem jeziku Java in tečejo na prilagojeni Dalvik virtualni napravi [10].

SQLite

SQLite je lahka podatkovna baza, ki ne potrebuje strežnika za svoje delovanje, ampak lahko piše neposredno na trdi disk. Vsa podatkovna baza, z vsemi tabelami je vsebovana v eni sami datoteki. Ta datoteka je tudi prenosljiva z enega operacijskega sistema na drugega. Prav tako ni občutljiva na razliko med 32 in 64 bitnimi sistemi in arhitekturo debelega in tankega konca. Prav zato je SQLite tako priljubljena in množično uporabljena podatkovna baza za shranjevanje podatkov. [11].

Android Emulator

Za testiranje aplikacije na osebnem računalniku je bil uporabljen Android emulator, ki je del Android SDK. To je virtualna mobilna naprava, ki teče na računalniku in nam s simulacijo Android okolja omogoča testiranje aplikacij brez fizične naprave [12].

2.3.2 Vertabelo

Za vizualno predstavo podatkovne baze je bila uporabljena spletna aplikacija *Vertabelo*, ki omogoča enostavno risanje tabel za različne tipe podatkovnih baz [13].

2.3.3 Strojna opema

Za testiranje aplikacije so bile uporabljene naslednje naprave:

- Sony Xperia SP,
- Samsung Galaxy Tab 2,
- Samsung Google Nexus S in

- Google Nexus 10.

Sony Xperia SP

Specifikacije naprave [14]:

- Zaslon: 4,6 inch (11,7 cm),
- Resolucija: 720 × 1280 slikovnih točk,
- Procesor: dvojedrni 1,7 GHz,
- Delovni spomin: 1024 MB RAM,
- NFC: da

Samsung Galaxy Tab 2

Specifikacije naprave [15]:

- Zaslon: 10,1 inch (25,7 cm),
- Resolucija: 1280 × 800 slikovnih točk,
- Procesor: dvojedrni 1,0 GHz,
- Delovni spomin: 1024 MB RAM,
- NFC: ne

Samsung Google Nexus S

Specifikacije naprave [18]:

- Zaslon: 4,0 inch (10,1 cm),
- Resolucija: 480 × 800 slikovnih točk,
- Procesor: enojedrni 1,0 GHz,
- Delovni spomin: 512 MB RAM,
- NFC: da

Google Nexus 10

Specifikacije naprave [19]:

- Zaslon: 10,1 inch (25,7 cm),
- Resolucija: 2560 × 1600 slikovnih točk,
- Procesor: dvojedrni 1,7 GHz,
- Delovni spomin: 2048 MB RAM,
- NFC: da

Poglavje 3

Preverjanje prisotnosti študentov z NFC izkaznicami

3.1 Opis problema

Problem, ki bi ga radi rešili, je avtomatizacija preverjanja prisotnosti študentov na vajah in izpitih. Poleg tega pa bi radi profesorjem omogočili tudi preprost način vpisovanja ocen ter uvoza in izvoza podatkov o študentih.

Za rešitev zastavljene naloge imamo na voljo študentske izkaznice NFC, s katerih je zaradi šifriranja podatkov mogoče prebrati samo UID izkaznice ter spletni servis STUDIS in `.csv` datoteke, preko katerih lahko pridobimo podatke o študentih.

Spletni servis STUDIS

Spletni servis STUDIS, ki ga ima aplikacija na voljo za uvoz podatkov, je bil razvit prav s tem namenom. Omogoča pridobivanje seznamov študentov, ki so prijavljeni na določen predmet ali izpitni rok. Podatki, ki jih aplikacija pridobi iz tega sistema so:

- UID izkaznice študenta,
- vpisna številka študenta,
- priimek in

- ime študenta ter
- število polaganj izpita (samo v primeru izpita).

CSV datoteke

Za uvoz podatkov imamo na razpolago tudi `.csv` datoteke, ki jih lahko profesor izvozi iz spletnega sistema STUDIS. Podatki v tej datoteki so:

- UID izkaznice študenta (predvideno po testni fazi prototipa),
- vpisna številka študenta,
- priimek in
- ime študenta ter
- število polaganj izpita (samo v primeru izpita).

NFC študentske izkaznice

Študentske izkaznice, ki jih bo odčitavala aplikacija, imajo vgrajeno pasivno NFC napravo modela MF3ICD81 proizvajalca NXP [8]. Te naprave so tipa ISO/IEC 14443-A in imajo kapaciteto 8KB. Vsaka ta značka ima unikaten identifikator (v nadaljevanju UID), ki je štirinajst mestno šestnajstiško število. UID je zaradi varnostnih razlogov programsko zapisan v del spomina, katerega je mogoče samo brati. To pomeni, da je to vrednost naknadno nemogoče spremeniti [8]. Na izkaznicah pa so zapisani še drugi podatki, ki vključujejo:

- šifra univerze,
- vpisna številka,
- priimek in
- ime študenta.

Za varnost osebnih podatkov je na napravi poskrbljeno z enkripcijo podatkov, za katere potrebujemo pravilni ključ, da jih lahko dešifriramo.

Iz podatkov je razvidno, da bo aplikacija sposobna povezovati študentske izkaznice s študenti s pomočjo unikatnih identifikatorjev izkaznic (UID), saj bo prejela te vrednosti ob uvozu seznamov in nato z odčitavanjem izkaznic.

3.2 Predlagana rešitev

Rešitev bo izdelana kot mobilna aplikacija za naprave z operacijskim sistemom Android. Ker bi bila lahko aplikacija te vrste tudi zlorabljena, bo zaščitena z geslom, ki si ga bo uporabnik izbral ob prvem zagonu aplikacije. V aplikacijo bo mogoče uvoziti sezname študentov s spletno storitvijo STUDIS in `.csv` datotekami in jih nato pregledovati. Za vsak seznam bo mogoče preverjati prisotnost študentov z NFC izkaznicami ali neposrednim vnašanjem v grafičnem vmesniku, ter jim dodajati ocene in komentarje. Profesor si bo lahko ustvaril več seznamov, za vsak predmet posebej, s katerimi bo nato lahko preverjal prisotnost. Aplikacija bo omogočala tudi izvoz seznamov z vsemi ocenami, prisotnostjo, komentarji in izračunano končno oceno. Seznami se bodo delili na dva tipa.

Izpit

Profesor bo moral v aplikacijo uvoziti podatke o študentih (UID izkaznice, vpisna številka, priimek, ime in število dosedanjih polaganj izpita). Aplikacija bo sprejemala datoteke tipa `.csv` ali sezname, ki jih vrača sistem STUDIS. Po uvozu bo imel možnost določanja študentove prisotnosti, njegove pisne in ustne ocene ter dodajanja komentarjev. Po končanem vpisovanju ocen in komentarjev, bo imel profesor na voljo izvoz seznama v obliki `.csv` datoteke.

Vaje

Za to vrsto seznama bo prav tako potrebno uvoziti podatke o študentih (UID izkaznice, vpisna številka, priimek in ime). Prav tako bo tudi v tem primeru možno uvažati sezname kot `.csv` datoteke ali preko sistema STUDIS. Profesor bo nato lahko študentom dodajal poljubno število ocen, ki jih bodo dosegali na vajah. Imel bo tudi pregled nad študentovo prisotnostjo in možnost dodajanja komentarjev. Na koncu bo imel profesor možnost seznam tudi izvoziti kot `.csv` datoteko.

3.3 Implementacija

Aplikacijo sestavlja aplikativni del in podatkovna baza. Aplikativni del skrbi za grafični vmesnik, povezovanje na sistem STUDIS, uvažanje in izvažanje seznamov ter za branje NFC izkaznic. Podatkovna baza pa je zadolžena za hranjenje podatkov. Vanjo se shranjujejo vsi uvoženi seznami študentov in podrobnosti o študentih.

3.3.1 Uvoz seznama

Uvoz seznama je možen samo z datotekami tipa `.csv` in preko sistema STUDIS. Če so seznami uvoženi preko sistema STUDIS, uporabniku ni potrebno skrbeti za pravilno strukturo seznama, ki jo pridobi aplikacija, saj za to poskrbi sistem sam. Aplikacija prejme od sistema STUDIS JSON objekt, ki vsebuje podatke o študentih. Glede na tip seznama aplikacija prejme različno zgrajene objekte.

Izpit

V primeru izpita je JSON objekt sestavljen iz izpitnih rokov in študentov prijavljenih na posamezen izpitni rok. Podatki o študentih, ki jih prejme so UID izkaznice, vpisna številka, priimek, ime in število polaganj.

Vaje

Če gre za vaje, aplikacija prejme samo en seznam. V tem seznamu so vsi študentje, ki so prijavljeni na željen predmet v tekočem študijskem letu. Podatki v seznamu so UID izkaznice, vpisna številka, priimek in ime.

Pri uvozih s `.csv` datotekami pa je potrebno biti pozoren na vsebino same datoteke. Vrstni red stolpcev v datoteki je zelo pomemben, saj aplikacija pričakuje podatke v določenem vrstnem redu. Glede na tip seznama, ki ga želimo uvoziti, aplikacija pričakuje drugačen vrstni red podatkov. To izvira iz načina, na katerega profesorju STUDIS izvozi podatke o študentih. Glede na to ali profesor izvaža datoteko s študenti, prijavljenimi na izpit, ali datoteko z vsemi študenti, vpisanimi na njegov predmet, se vrstni red podatkov in sami podatki nekoliko razlikujejo.

Spletne tehnologije 2013/14, 17.6.2014 - seznam prijavljenih na izpitni rok				
04757B7AFC2E80	63111111	NOVAK, KRISTJAN	1	
040A5782FC2E80	63111222	GOLOB, GAŠPER	1	
0497647AFC2E80	63111333	DOLINAR, LUKA	2	

Slika 3.1: Izgled pravilne `.csv` datoteke za uvoz seznama tipa *izpit*.

63727 Spletne tehnologije 2013/14	- seznam študentov izvajanja			
UID	Vpisna številka	Priimek, Ime	Letnik program	E-mail
04757B7AFC2E80	63111111	NOVAK, KRISTJAN	3 BVS-RI	7243@naslov.si
040A5782FC2E80	63111222	GOLOB, TADEJ	3 BVS-RI	9344@naslov.si
0497647AFC2E80	63111333	DOLINAR, ŽAN	2 BVS-RI	0748@naslov.si

Slika 3.2: Izgled pravilne `.csv` datoteke za uvoz seznama tipa *vaje*.

Izpit

Pri uvažanju seznama za izpit aplikacija pričakuje v prvi vrstici podatke o predmetu, nato podatke o vsakem študentu v svoji vrstici, s stolpci UID izkaznice, vpisna številka, priimek in ime (ločena z vejico) ter število polaganj (slika 3.1). Vrstni red stolpcev je pomemben. Pomembno je tudi, da so stolpci med seboj ločeni z belimi presledki in ne s piko ali vejico. Če ima datoteka še druge stolpce, to ni pomembno. Pomembno je samo da so za naštetimi stolpci.

Vaje

V primeru uvažanja seznama, ki bo uporabljen za preverjanje prisotnosti na vajah, pa aplikacija pričakuje v prvi vrstici podatke o predmetu, v drugi imena stolpcev in nato podatke vsakega študenta v svoji vrstici, s stolpci UID izkaznice, vpisna številka, priimek in ime (slika 3.2). Priimek in ime morata biti v enem stolpcu in ločena z vejico. Vrstni red stolpcev je pomemben. Prav tako je pomembno tudi, da so stolpci ločeni z belimi presledki in ni pomembno, ali so v datoteki dodani še drugi stolpci, v kolikor so za omenjenimi.

3.3.2 Podatkovni model

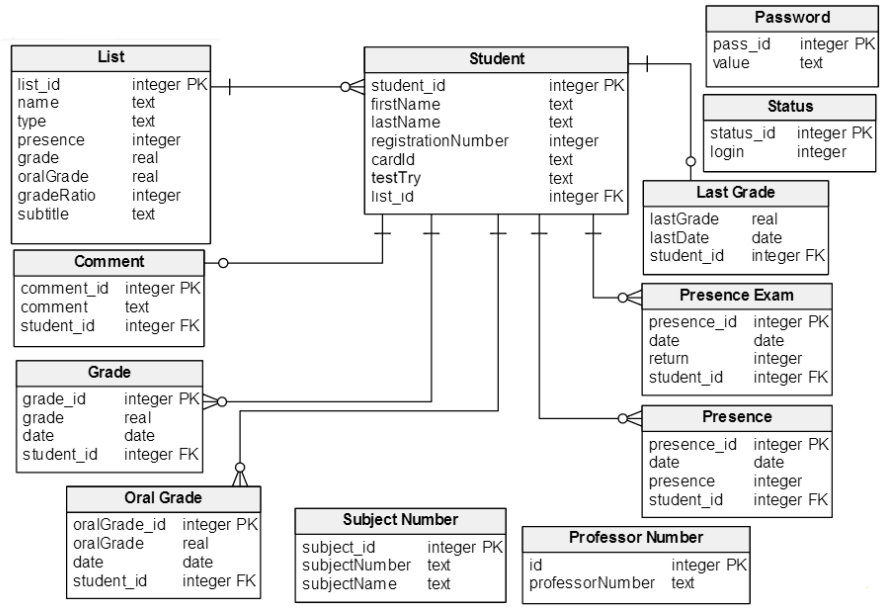
Podatkovni model aplikacije mora biti sposoben hraniti več seznamov študentov, saj lahko profesor predava tudi več kot samo en predmet. Vsak seznam (ang. *list*)

mora imeti določen tip (ang. *type*), ali gre za vaje, ali za izpit, ime seznama (ang. *name*) in podnaslov (ang. *subtitle*), ki ga uporabnik ne izbere sam, ampak se prebere iz uvožene `.csv` datoteke oziroma s pridobljenih podatkov preko STUDIS spletnega servisa. Glede na tip seznama pa je potrebno hraniti v tabeli še minimalno pozitivno oceno pisnega izpita, minimalno pozitivno oceno ustnega izpita in razmerje omenjenih ocen, za izračun končne ocene, v primeru, da gre za izpit. V primeru, da gre za vaje pa je potrebno hraniti minimalno prisotnost študenta in minimalno pozitivno končno oceno za uspešno opravljene vaje. To je prva tabela aplikacije. Poimenovana je *List* in vsebuje naslednje stolpce[1] (slika 3.3):

1. **ID** (*list_id*) - primarni ključ,
2. **IME** (*name*) - ime uvoženega seznama,
3. **TIP** (*type*) - tip seznama,
4. **PRISOTNOST** (*presence*) - minimalna prisotnost, potrebna za uspešno opravljene vaje,
5. **OCENA** (*grade*) - minimalna ocena potrebna, za uspešno opravljene vaje ali izpit,
6. **USTNA OCENA** (*oralGrade*) - minimalna ocena potrebna za uspešno opravljen ustni izpit,
7. **RAZMERJE OCEN** (*gradeRatio*) - vrednost pisnega izpita proti ustnemu (med 1 in 100),
8. **PODNASLOV** (*subtitle*) - prebran iz uvožene `.csv` datoteke oziroma podatkov, pridobljenih s STUDIS spletnega servisa (izpiše se pod imenom seznama).

Vsak seznam, ki ga bo uporabnik uvozil, bo imel vsaj enega ali več študentov. Zato mora biti aplikacija sposobna hraniti tudi več študentov (ang. *student*) in jih ustrezno dodeliti seznamom. Za vsakega študenta je potrebno hraniti njegovo ime (ang. *firstname*), priimek (ang. *lastname*), vpisno številko (ang. *registration number*), UID študentske izkaznice, število polaganj izpita ter tuji ključ tabele *list*,

[1] Vsi stolpci in tabele v aplikaciji so poimenovane v angleščini.



Slika 3.3: Podatkovna baza in relacije med tabelami.

da bo študenta mogoče povezati z ustreznim seznamom. Za hranjenje študentov aplikacija uporablja tabelo *Student*. Tabela vsebuje naslednje stolpce (slika 3.3):

1. **ID** (*student_id*) - primarni ključ,
2. **IME** (*firstName*) - ime študenta,
3. **PRIIMEK** (*lastName*) - priimek študenta,
4. **VPISNA ŠTEVILKA** (*registrationNumber*) - vpisna številka študenta,
5. **UID IZKAZNICE** (*cardId*) - unikatni identifikator študentove izkaznice,
6. **ŠTEVILO POLAGANJ** (*testTry*) - število polaganj izpita (v primeru seznama tipa *izpit*),
7. **ID SEZNAMA** (*list_id*) - identifikator, ki povezuje študenta s seznamom.

Ker je namen te aplikacije preverjanje prisotnosti, je potrebno beležiti tudi prisotnost. Glede na tip seznama, v katerem je shranjen študent, je potrebno hraniti različne podatke. Za izpit je potrebno hraniti datum prisotnosti, tuji ključ

študenta in ali je bila prijava študentu vrnjena. Za vaje pa je potrebno hraniti datum zadnje prisotnosti, števec, ki nam pove, kolikokrat je bil študent prisoten na vajah in tuji ključ študenta. Zaradi teh razlik aplikacija uporablja dve tabeli za beleženje prisotnosti glede na tip seznama. Tabela za prisotnost na vajah je poimenovana *Presence*, tabela za izpite pa *PresenceExam*. Stolpci v tabeli *Presence* (slika 3.3):

1. **ID** (*presence_id*) - primarni ključ,
2. **DATUM** (*date*) - hrani zadnji datum vaj, na katerih je bil študent prisoten,
3. **PRISOTNOST** (*presence*) - je števec, ki nam pove, kolikokrat je bil študent prisoten na vajah,
4. **ID ŠTUDENTA** (*student_id*) - identifikator, ki povezuje prisotnost s študentom.

Stolpci tabele prisotnosti za izpit pa so (slika 3.3):

1. **ID** (*presence_id*) - primarni ključ,
2. **DATUM** (*date*) - je zapisan v primeru, da je študent prišel na izpit,
3. **VRNJENA PRIJAVA** (*return*) - nam pove, ali je profesor vrnil prijavo študentu,
4. **ID ŠTUDENTA** (*student_id*) - identifikator, ki poveže prisotnost s študentom.

Ena od funkcij aplikacije je tudi možnost vpisa ocen. Za oceno je potrebno hraniti datum, oceno in tuji ključ študenta. Kot pri prisotnosti, se tudi tukaj delovanje aplikacije razlikuje glede na tip seznama. Aplikacija hrani pisne in ustne ocene ločeno vsake v svoji tabeli. Čeprav je potrebno hraniti enake podatke tako za pisno, kot tudi za ustno oceno, je potrebno tudi ločiti eno od druge. Tabeli, ki ju aplikacija uporablja za ta namen, se imenujeta *grade* in *oralGrade*. V primeru, da je seznam tipa izpit, se uporabita obe tabeli, v nasprotnem primeru pa samo tabela *Grade*. Stolpci v tabeli *Grade* (slika 3.3):

1. **ID** (*grade_id*) - primarni ključ,
2. **OCENA** (*grade*) - pisna ocena, ki jo je dosegel študent,

3. **DATUM** (*date*) - datum dosežene ocene,

4. **ID ŠTUDENTA** (*student_id*) - identifikator, ki poveže študenta z oceno.

Stolpci v tabeli *oralGrade* (slika 3.3):

1. **ID** (*oralGrade_id*) - primarni ključ,

2. **OCENA** (*oralGrade*) - ustna ocena, ki jo je dosegel študent,

3. **DATUM** (*date*) - datum dosežene ocene,

4. **ID ŠTUDENTA** (*student_id*) - identifikator, ki poveže študenta z oceno.

Za komentarje, ki jih profesor lahko doda vsakemu študentu, aplikacija uporablja posebno tabelo, ki se imenuje *Comment*. Pri komentarju je potrebno hraniti njegovo vsebino in tuji ključ študenta, da ga lahko povežemo z njim. Zato so stolpci v tej tabeli naslednji (slika 3.3):

1. **ID** (*comment_id*) - primarni ključ,

2. **KOMENTAR** (*comment*) - vsebina komentarja, ki jo vnese profesor,

3. **ID ŠTUDENTA** (*student_id*) - identifikator, ki komentar poveže s študentom.

Za lažje dostopanje do zadnjih študentovih ocen in prisotnosti, aplikacija uporablja dodatno tabelo. Ta tabela v primeru, da gre za študenta s seznama tipa *izpit*, hrani seštevek ocen, doseženih na ustnem in pisnem izpitu ter datum v primeru, da je študent na izpit prišel. V primeru, da gre za študenta s seznama tipa *vaje*, pa hrani zadnjo oceno vaj, ter datum zadnje prisotnosti na vajah. Tabela se imenuje *LastGrade* in ima sledeče stolpce (slika 3.3):

1. **ZADNJA OCENA** (*lastGrade*) - zadnja ocena z vaj oziroma skupna ocena z izpita,

2. **ZADNJI DATUM** (*lastDate*) - zadnji datum, ko je bil študent prisoten na vajah oziroma datum, ko je bil študent prisoten na izpitu,

3. **ID ŠTUDENTA** (*student_id*) - tuji ključ študenta.

Za shranjevanje profesorjeve šifre, aplikacija uporablja tabelo *ProfessorNumber*. Ta mora hraniti samo šifro profesorja, zato je dokaj trivialna. Njena stolpca sta (slika 3.3):

1. **ID** (*id*) - primarni ključ,
2. **ŠIFRA PROFESORJA** (*professorNumber*) - šifra profesorja.

Uporabnik ima možnost dodajanja predmetov, ki jih predava. Za hranjenje teh podatkov ima podatkovna baza tabelo *SubjectNumber*, v kateri so shranjeni predmeti, ki jih uporabnik vnese. Njeni stolpci so (slika 3.3):

1. **ID** (*subject_id*) - primarni ključ,
2. **ŠIFRA PREDMETA** (*subjectNumber*) - šifra predmeta,
3. **IME PREDMETA** (*subjectName*) - naziv shranjenega predmeta.

Ker je aplikacija zaščitena z geslom, ki si ga uporabnik izbere ob prvem zagonu, je potrebno tudi geslo hraniti. Za hranjenje gesla se uporablja tabela z imenom *Password*. V tej tabeli je vedno samo ena vrstica, saj ni potrebno hraniti več kot enega gesla. Stolpca te tabele sta (slika 3.3):

1. **ID** (*pass_id*) - primarni ključ,
2. **GESLO** (*value*) - uporabniško geslo.

Za avtentikacijo uporabnika aplikacija uporablja tabelo, ki hrani vrednost ali je uporabnik prijavljen ali odjavljen. Tudi pri tej tabeli bo vedno samo ena vrstica, saj ima uporabnik v vsakem trenutku samo eno stanje. Imenuje se *Status*, njena stolpca pa sta (slika 3.3):

1. **ID** (*status_id*) - primarni ključ,
2. **PRIJAVLJEN** (*login*) - nam pove, ali je uporabnik trenutno prijavljen.

3.3.3 Arhitektura aplikacije

Aplikacija za svoje delovanje potrebuje napravo z Android operacijskim sistemom *Ice cream sandwich* (verzija 4.0) ali novejšim. Ta verzija operacijskega sistema je

bila izbrana, ker je to najnižja verzija, ki že podpira NFC tehnologijo. V aplikacijo je mogoče ročno vnašati prisotnost študentov, ocen in komentarjev. Za te funkcionalnosti NFC ni potreben. Aplikacija je sestavljena iz štirih aktivnosti (ang. *activity*):

- *SeznamListActivity*,
- *SeznamDetailActivity*,
- *FileSelectionActivity* in
- *NfcPresenceCheck*

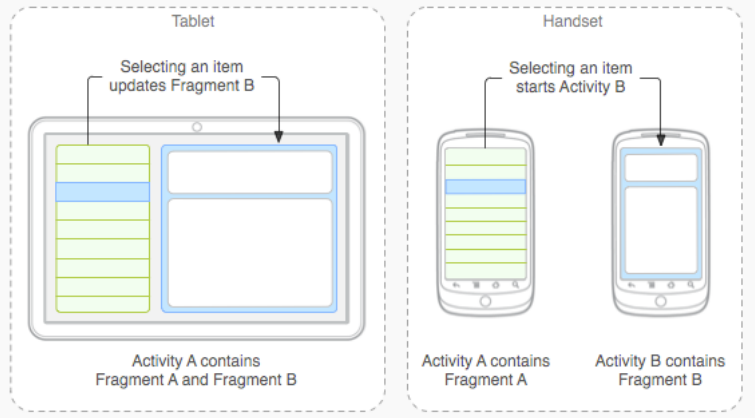
ter dveh fragmentov (ang. *fragments*):

- *SeznamListFragment* in
- *SeznamDetailFragment*.

Ob zagonu aplikacije se najprej odpre aktivnost, ki prikazuje vse ustvarjene sezname (*SeznamListActivity*). Ta aktivnost ne izgleda na vseh napravah enako. Izgled aktivnosti je odvisen od velikosti zaslona naprave. Na manjših zaslonih se prikaže s fragmentom za prikaz vseh uvoženih seznamov (*SeznamListFragment*). Na večjih zaslonih pa se poleg omenjenega fragmenta prikaže tudi fragment za prikazovanje študentov z izbranega seznama (*SeznamDetailFragment*). Tako lahko na večjih zaslonih vidimo hkrati vse svoje uvožene sezname na levi in študente iz izbranega seznama na desni strani zaslona. V tem primeru se aktivnost za prikaz študentov z izbranega seznama (*SeznamDetailActivity*) nikoli ne prikaže. V primeru manjšega zaslona se ta aktivnost prikaže, ko izberemo željen seznam. Delovanje fragmentov je prikazano na sliki 3.4.

Tretja aktivnost (*FileSelectionActivity*) se prikaže, ko želi uporabnik uvoziti nov `.csv` seznam. Zadnja aktivnost (*NfcPresenceCheck*), pa je aktivnost, ki se odpre ob preverjanju prisotnosti z NFC izkaznicami. Ta aktivnost se bo odprla samo, če ima naprava, na kateri poganjamo aplikacijo, podprto NFC tehnologijo.

Za vnašanje podatkov o seznamih, dodajanje ocen, prisotnosti in komentarjev niso bile uporabljene aktivnosti, ampak dialogi (ang. *dialog*). Dialogi omogočajo prikaz kar znotraj aktivnosti in je ni potrebno začasno zaustaviti ali zapirati.



Slika 3.4: Prikaz aplikacije na velikih in majhnih zaslonih.

3.3.4 Predloga 'Master/Detail flow'

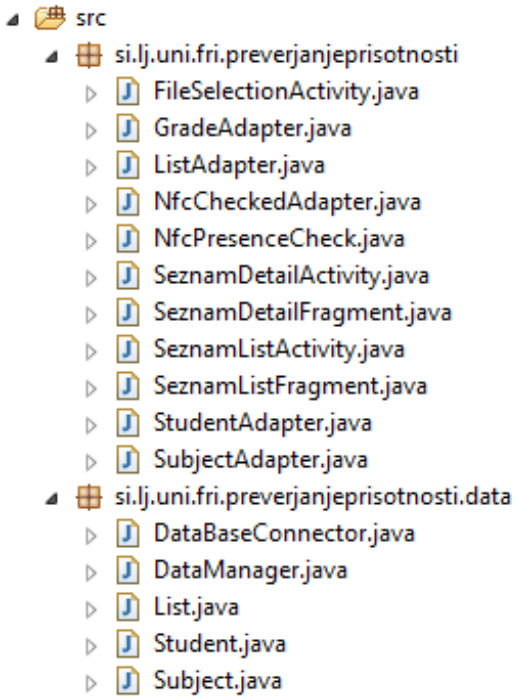
Zaradi lepšega prikaza na vseh velikostih zaslona aplikacija uporablja prej omenjene fragmente. Android SDK nam ponuja že vnaprej pripravljeno predlogo (ang. *template*) za tak pogled. Ta predloga je poimenovana *Master/Detail flow*. Omogoča nam prikaz seznama uvoženih seznamov na glavnem delu zaslona (ang. *master*), ter dodatne informacije izbranega seznama na zaslonu za podrobnosti (ang. *detail*) [16].

3.3.5 Razdelitev aplikacije

Aplikacija je razdeljena na dva paketa (ang. *package*) (slika 3.5):

- *si.lj.uni.fri.preverjanjeprisotnosti*
- *si.lj.uni.fri.preverjanjeprisotnosti.data*.

Taki imeni sta bili izbrani, ker morajo biti paketi Android poimenovani unikatno. V prvem paketu so vse aktivnosti, fragmenti ter adapterji seznamov (ang. *list adapter*). Drugi paket pa je zadolžen za upravljanje s podatki. Tu imamo razred, ki ustvari podatkovno bazo, razrede, ki predstavljajo objekte (sezname, študente in predmete), ter razred, ki ustvari sezname za prikaz na glavnem zaslonu.



Slika 3.5: Paketi v aplikaciji.

Aplikativni paket

Aktivnost, ki se zažene prva ob zagonu aplikacije je *SeznamListActivity*. Glede na velikost zaslona se prikaže skupaj z *SeznamListFragment* fragmentom za manjše zaslone, oziroma z omenjenim fragmentom in *SeznamDetalFragment* fragmentom za večje zaslone. *SeznamListActivity* definira dialog za vnos gesla ob zagonu aplikacije. Določa tudi meni z nekaj možnostmi in akcije, ki se zgodijo ob pritisku na izbire. Ob pritisku na enega od prikazanih seznamov zažene novo aktivnost oziroma fragment, v primeru velikega zaslona in mu posreduje unikatni identifikator izbranega seznama.

SeznamListFragment je fragment, ki se vedno zažene z omenjeno aktivnostjo. Ta razred določa meni, ki se pojavi ob dolgem pritisku na željen seznam in določa akcije, ki se zgodijo glede na izbrano opcijo z menija.

Drugi fragment je *SeznamDetailFragment*. Ta se zažene s *SeznamListActivity* aktivnostjo na večjih zaslonih in s *SeznamDetailActivity* aktivnostjo na manjših zaslonih. Ta razred pridobi unikaten identifikator izbranega seznama, ki ga prejme preko namena in prikaže študente, ki so na izbranem seznamu. Študente prikaže v obliki seznama in poskrbi, da je vsakega od njih možno izbrati. Ob izbiri enega od študentov poskrbi za prikaz dialoga s študentovimi podatki in do sedaj vnesenimi vrednostmi.

Aktivnost *SeznamDeatilActivity* se zažene samo na napravah z manjšimi zasloni in je zadolžena za meni, ki je identičen meniju, ki ga ponuja *SeznamListFragment*. Ta razred definira tudi akcije, ki se zgodijo ob izbiri, katere od opcij s tega menija.

Naslednja aktivnost, ki se požene ob uvozu datoteke, je *FileSelectionActivity*. Ta nam omogoča prikaz datotečnega sistema naše naprave in izbiro datoteke za uvoz. Izbrano datoteko nato posreduje aktivnosti *SeznamListActivity*, ki uvozi študente.

Zadnja aktivnost je poimenovana *NfcPresenceCheck* in je zadolžena za branje NFC študentskih izkaznic. Tudi ta aktivnost ima svoj meni in določa akcijo, ki jo proži izbira opcije v njem.

Ostali razredi v tem paketu so adapterji za sezname. Aplikacija namreč prikazuje pet seznamov. To so:

- seznam uvoženih seznamov,

Slika 3.6: Elementi seznamov *ListAdapter, StudentAdapter, NfcCheckedAdapter, GradeAdapter in SubjectAdapter* (od zgoraj navzdol).

- seznam študentov izbranega seznama,
- seznam študentov, katerih prisotnost smo že preverili,
- seznam ocen študenta in
- seznam shranjenih predmetov, ki jih predava profesor.

Vsak od teh adapterjev, *ListAdapter, StudentAdapter, NfcCheckedAdapter, GradeAdapter in SubjectAdapter*, določa izgled enega elementa v seznamu (slika 3.6).

Podatkovni paket

Eden od razredov v tem paketu je zadolžen za opravljanje s podatkovno bazo. V tem razredu se nahaja struktura podatkovne baze, ki se ustvari ob prvem zagonu naše aplikacije in vse metode, ki pišejo v podatkovno bazo, ali berejo z nje. Poimenovan je *DatabaseConnector*.

Drugi razred, je razred *DataManager*. Ta je zadolžen za pripravo že uvoženih seznamov, ki jih bo prikazala aplikacija ob zagonu. Razred je bil že del *Master/Detail flow* predloge, vendar ga je bilo potrebno prilagoditi, saj je predloga prikazovala statično vsebino in ne vsebino iz podatkovne baze. Ta razred je definiran za dodajanje novega in odstranjevanje obstoječega seznama, ter dodajanje manjkajočih seznamov. Zadnja funkcionalnost je potrebna zaradi načina delovanja *Master/Detail flow* predloge. Ta si obstoječe sezname, ki jih prikazuje, zapomni tudi ob izhodu iz aplikacije, vendar če bi aplikacijo prisilno zaustavili, se ta spomin izgubi. Zato je potrebno ob zagonu aplikacije vedno preverili, če kateri od seznamov, ki je shranjen v podatkovni bazi, ni prikazan in ga po potrebi prikazati.

Preostali razredi so *List*, *Student* in *Subject*. Ti razredi predstavljajo objekte, ki jih razred *DatabaseConnector* vrača ob poizvedovanju po podatkovni bazi. Sestavljeni so iz konstruktorja in metod za nastavljanje in pridobivanje podatkov.

3.3.6 Šifriranje uporabniškega gesla

Gesla ni nikoli pametno shranjevati kot nešifrirano besedilo, saj bi bilo ob napadu na podatkovno bazo, ki vsebuje gesla, za napadalca zelo enostavno pridobiti vsa shranjena gesla in jih zlorabiti. Prav zato aplikacija uporablja šifriranje gesla, ki si ga je uporabnik izbral ob prvem zagonu aplikacije.

Za šifriranje gesla se uporablja algoritem SHA (*Secure hash algorithm*), ki je že na voljo v programskem jeziku Java. Ta algoritem je po navadi uporabljen tudi v varnostnih protokolih, kot so PGP (ang. *pretty good privacy*), TLS (ang. *transport layer security*), SSH (ang. *secure shell*) in SSL (ang. *secure socket layer*) [17].

3.3.7 Izvoz seznama

Aplikacija ima tudi možnost izvoza seznama z vsemi vpisanimi ocenami in prisotnostmi. Za izvožene sezname bo aplikacija na napravi ustvarila novo mapo imenovano *PreverjanjePrisotnosti-Izvozi* in shranila vse izvožene sezname v to mapo. Seznami, ki jih izvozi aplikacija se razlikujejo glede na tip.

Izpit

Za izpit aplikacija izvozi seznam s stolpci v naslednjem vrstnem redu:

1. **vpisna številka**,
2. **priimek**,
3. **ime**,
4. **prisotnost**,
5. **pisna ocena**,
6. **ustna ocena** in
7. **končna ocena**.

Prisotnost je označena z 1, če je bil študent prisoten na izpitu, z 0, če ga ni bilo in z VP v primeru, da je profesor študentu prijavo vrnil. Oblika `.csv` datoteke, ki jo izvozi aplikacija, je oblikovana tako, da jo razume tudi STUDIS sistem. Ta namreč za uvoz ocen potrebuje v prvem stolpcu vpisno številko in v zadnjem končno oceno.

Vaje

V primeru, da gre za vaje aplikacija izvozi seznam z naslednjimi stolpci:

1. **vpisna številka**,
2. **priimek**,
3. **ime**,
4. **prisotnost**,
5. **ocene in skupna ocena**.

Prisotnost je izražena s številom, glede na število vaj, na katerih je bil študent prisoten. Stolpec *ocene in skupna ocena*, pa je v resnici več stolpcev. Vsaka ocena, ki jo profesor vnese za določenega študenta, se izpiše v nov stolpec. V zadnji stolpec pa se izpiše povprečna ocena študenta.

3.3.8 Branje NFC študentskih izkaznic

Študentske izkaznice odčitavajo razredi *SeznamListActivity*, *SeznamDetailActivity* in *NfcPresenceCheck*. Te aktivnosti potrebujejo identifikator seznama, za katerega bo uporabnik pregledoval prisotnost. *SeznamDetailActivity* in *NfcPresenceCheck* ga pridobita preko namena, ki ga ustvari *SeznamListActivity* (v prvem in drugem primeru) ali *SeznamDetailActivity* (v drugem primeru). V primeru, da tega parametra ni, se aplikacija preusmeri na začetno aktivnost, ali pa uporabnika obvesti, da je potrebno najprej izbrati seznam. Kaj se zgodi, je odvisno od tega, kje v aplikaciji se nahajamo.

Vse aktivnosti najprej preverijo prisotnost NFC tehnologije v napravi. V kolikor NFC tehnologija ni podprta *SeznamListActivity* in *SeznamDetailActivity* tečeta naprej normalno, saj lahko uporabnik še zmeraj uporablja druge funkcionalnosti, *NfcPresenceCheck* pa se zaustavi. Za delovanje tega dela aplikacije je namreč nujno potrebna NFC tehnologija, saj ne opravlja nobene druge funkcionalnosti, kot preverjanje prisotnosti z NFC tehnologijo. V primeru, da je naprava ne podpira, se ta aktivnost zapre in obvesti uporabnika, da ta funkcionalnost ni podprta na njegovi napravi. Aktivnost se prav tako zapre, če uporabnikova naprava podpira NFC tehnologijo, a je ta izključena. Takrat aplikacija uporabnika opomni, naj priklopi svojo NFC napravo na telefonu.

Če je vsem omenjenim pogojem zadoščeno, aktivnost pridobi vse študente, katerih prisotnost je bila že zabeležena. Pri vseh treh aktivnostih je potrebno na začetku nastaviti sistem za prioritetno odpremo značk (ang. *Foreground dispatch system*). To pomeni, da bo ob približanju NFC značke aplikacija prevzela prioriteto za obdelavo namena, ki ga bo ustvaril sistem za odpremo značk (ang. *Tag dispatch system*). To je potrebno nastaviti vsakič, ko se aktivnost zažene ali prebudi, v primeru, da je bila začasno zaustavljena. Zato je metoda, ki opravlja to nalogo v vseh aktivnostih klicana v metodi *onResume*, ki se izvede vsakič, ko pride aktivnost v ospredje. Prav tako, kot je potrebno ob vsakem zagonu aktivnosti nastaviti sistem za prioritetno odpremo značk, ga je potrebno tudi ob zapiranju ali začasni zaustavitvi aktivnost onemogočiti. V primeru, da tega ne storimo, bi naša aplikacija kljub temu, da ni v ospredju, zahtevala prioriteto nad morebitnimi nameni, ki se ustvarijo ob branju NFC značk. S tem bi prišlo do nepravilnega delovanja v napravi. Da se to ne bi dogajalo, je v funkciji *onPause*, ki se izvede

```
@Override
public void onResume(){
    super.onResume();
    // Disabling other applications to run on NFC tag close up
    setupForegroundDispatch(this, nfcAdapter);
}

@Override
public void onPause(){
    // Enabling to run other applications on NFC tag close up
    stopForegroundDispatch(this, nfcAdapter);
    super.onPause();
}

// Disabling other applications to run on NFC tag close up
public static void setupForegroundDispatch(final Activity activity, NfcAdapter nfcAdapter){
    final Intent intent = new Intent(activity.getApplicationContext(), activity.getClass());
    intent.setFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);

    final PendingIntent pendingIntent = PendingIntent.getActivity(activity.getApplicationContext(), 0, intent, 0);

    IntentFilter[] filters = new IntentFilter[2];
    String[][] techList = new String[][]{new String[]{NfcA.class.getName(), NfcV.class.getName()}};

    filters[0] = new IntentFilter();
    filters[0].addAction(NfcAdapter.ACTION_TECH_DISCOVERED);
    filters[1] = new IntentFilter();
    filters[1].addAction(NfcAdapter.ACTION_TAG_DISCOVERED);

    nfcAdapter.enableForegroundDispatch(activity, pendingIntent, filters, techList);
}

// Enabling other devices to run on NFC tag close up
public static void stopForegroundDispatch(final Activity activity, NfcAdapter nfcAdapter){
    nfcAdapter.disableForegroundDispatch(activity);
}
```

Slika 3.7: Implementacija sistema za prioritetno odpremo značk.

vsakič, ko aktivnost ni več v ospredju, klicana funkcija, ki onemogoči ta sistem (slika 3.7).

Študentske NFC izkaznice so naprave tipa A, kar pomeni, da je možno z njih brati in nanje ponovno pisati [5]. V sistemu za prioritetno odpremo značk je potrebno nastaviti, katere značke našo aplikacijo zanimajo. S slike 3.7 je razvidno, da je nastavljen filter za NFC naprave tipa A in V. Filter za značke tipa V je bil dodan, da bi bila aplikacija uporabna tudi za morebitne druge namene. Spremenljivka *techList* vsebuje namreč ime razredov NfcA in NfcV, ki sta razreda za NFC naprave tipa A in V.

Ko je sistem za prioritetno odpremo značk nastavljen, je aktivnost pripravljena za branje NFC značk. Ob približanju značke omenjenih tipov, se bo ustvaril namen in se posredoval direktno tej aplikaciji. Za obravnavo tega namena skrbi metoda *handleIntent* (slika 3.8) in je klicana v funkciji *onNewIntent*, ki se izvede vsakič, ko aktivnost prejme nov namen. Omenjena funkcijo prebere UID s približane izkaznice in ga razčleni tako, da dobi iz sprva prebranih zlogov (ang. *byte*), štirinajst mestno besedilo, ki je v resnici število v šestnajstiškem zapisu. Da je prebrana vrednost besedilo namesto števila ni popolnoma nič narobe, saj so v podatkovni bazi

```
@Override
public void onNewIntent(Intent intent){
    handleIntent(intent);
}

// Handling intent on NFC tag close up
protected void handleIntent(Intent intent){
    Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
    // Reading tag ID
    byte[] byteId = tag.getId();
    // Parsing ID from byte to string
    StringBuffer stringId = new StringBuffer();
    for (byte b : byteId){
        stringId.append(String.format("%02X", b));
    }
```

Slika 3.8: Metoda *onNewIntent* in del metode *handleIntent*, ki s študentske izkaznice prebere UID in ga razčleni v besedilo.

unikatni identifikatorji izkaznic prav tako shranjeni kot besedilo. Aktivnost nato prebrano vrednost, s pomočjo *DatabaseConnector* razreda, primerja z že prej prebranimi vrednostmi in z vrednostmi, ki jih imajo študentje seznama, za katerega pregledujemo prisotnost. Če študentove prisotnosti še nismo preverili, se študentu zabeleži prisotnost. Če smo njegovo prisotnost že preverili, nas aplikacija obvesti, da je študent že na seznamu prisotnih. V primeru, da aktivnost ne najde na seznamu študenta, ki bi imel študentsko izkaznico s takim identifikatorjem, pa nas aplikacija opozori, da študenta ni na tem seznamu. Na koncu sledi še posodobitev prikazanega seznama.

Poglavje 4

Testiranje in uporaba aplikacije

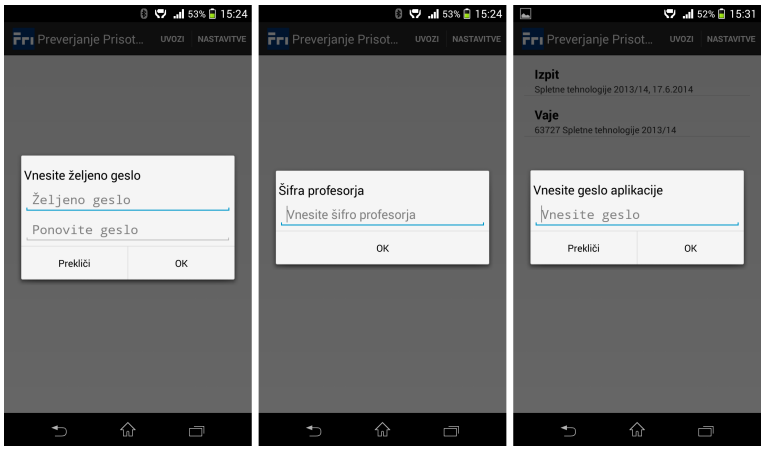
4.1 Testiranje

Za testiranje aplikacije je bil uporabljen *Android Emulator* in pametne naprave *Sony Xperia SP*, *Samsung Galaxy Tab 2*, *Google Nexus S* in *Google Nexus 10*. Vse naprave poganja operacijski sistem Android *Ice Cream Sandwich* (verzija 4.0) ali novejši. Na vsaki od naštetih naprav so bile testirane vse funkcionalnosti, razen preverjanja prisotnosti z NFC izkaznicami, saj NFC tehnologija ni vgrajena v *Samsung Galaxy Tab 2* napravo.

Aplikacija je bila nameščena na vse naprave. Na vsaki sta bila uvožena vsaj po dva seznama. Prvi seznam je bil tipa *izpit*, drugi pa tipa *vaje*. Testiran je bil izgled aplikacije na vseh napravah v pokončnem in ležečem načinu. Razen prikaza in preverjanja prisotnosti z NFC izkaznicami (*NfcPresenceCheck*), kjer naprava nujno potrebuje NFC, se delovanje ni razlikovalo na nobeni od naštetih naprav.

4.2 Uporaba

Ko aplikacijo prvič poženemo od nas zahteva, da si izberemo geslo in vnesemo šifro profesorja (slika 4.1). Če uporabnik vnese omenjene podatke pravilno, se odpre aplikacija in uporabnika hkrati tudi zabeleži kot prijavljenega. Od tega trenutka dalje, bo zagon aplikacije izgledal drugače, saj uporabniku ne bo potrebno ponovno izbirati gesla in vnašati šifre profesorja, ampak se bo v aplikacijo samo prijavil



Slika 4.1: Prvi zagon aplikacije (levo in na sredini), ter ponovni zagon aplikacije (desno).

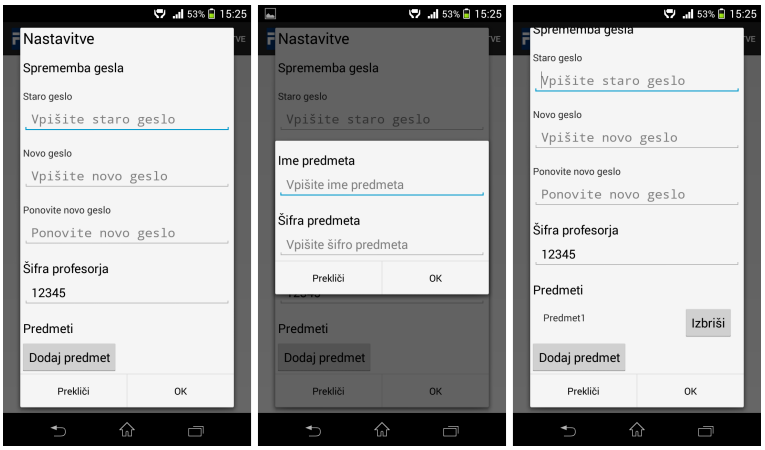
z geslom, ki si ga je že izbral (slika 4.1). Vnesene vrednosti lahko uporabnik kasneje še spreminja v nastavitvah aplikacija (slika 4.2). Tu ima možnosti menjave uporabniškega gesla, šifre profesorja in dodajanja predmetov, ki jih izvaja (slika 4.2).

Če uporabnik vstopa v aplikacijo prvič bo pred njim prazen zaslon (slika 4.3), drugače bodo prikazani že uvoženi seznami. V primeru praznega zaslona je potrebno uvoziti prvi seznam. To je možno storiti z gumbom *UVOZI* (slika 4.3).

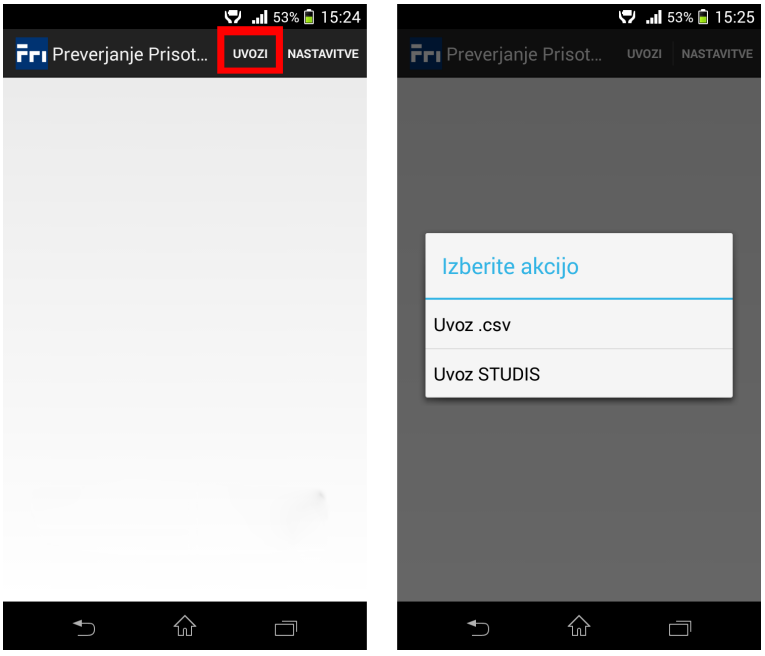
Ob pritisku na *UVOZI*, se odpre dialog (slika 4.3), ki uporabnika vpraša ali želi uvoziti seznam kot `.csv` datoteko, ki jo ima shranjeno na svoji napravi, ali želi seznam uvoziti preko sistema STUDIS.

- **Uvoz .csv**
  V primeru, da uporabnik izbere opcijo *Uvoz .csv*, se odpre nova aktivnost, ki omogoča izbiro datotek (slika 4.4). Ta aktivnost nam prikaže datoteke, ki so shranjene na naši napravi. V tem primeru je potrebno pred uporabo aplikacije naložiti ustrezne .csv datoteke na napravo.

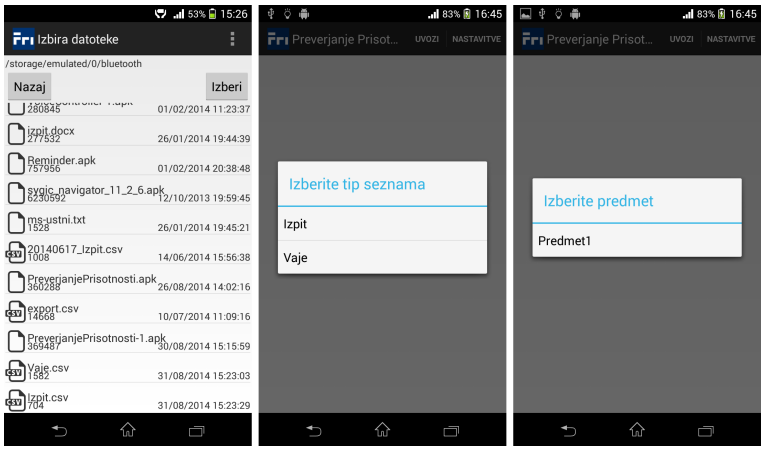
- **Uvoz STUDIS**
  Ob izbiri uvoza s sistemom *STUDIS*, se uporabniku odpre dialog, v katerem lahko izbere tip seznama. Nato se ponovno pojavi drugi dialog, kjer mora uporabnik izbrati enega od predmetov, ki jih izvaja. Če uporabnik ni vnesel



Slika 4.2: Nastavitve in postopek dodajanja predmetov.



Slika 4.3: Prazen začetni seznam in način uvažanja seznamov.



Slika 4.4: Izbira datoteke iz pomnilnika naprave (levo), izbira tipa seznama ob uvozu iz STUDIS sistema (na sredini) in izbira predmeta (desno).

še nobenega predmeta, se odpre dialog za vnos novega predmeta. Po končani izbiri predmeta mora uporabnik določiti še, katere seznam želi uvoziti, v primeru, da je izbral tip seznama *izpit* (slika 4.4).

Ko se uporabnik odloči, kateri seznam želi uvoziti, se prikaže dialog, v katerem je mogoče izbrati ime seznama. V kolikor uporabnik uvaža `.csv` datoteko, mora v tem koraku izbrati še tip seznama (slika 4.5). Aplikacija kot privzeto ime predlaga kar ime datoteke oziroma ime predmeta.

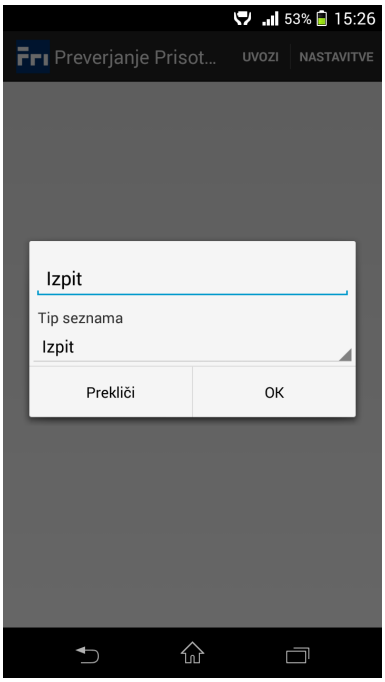
Tu se delovanje aplikacije loči glede na izbran tip seznama.

Izpit

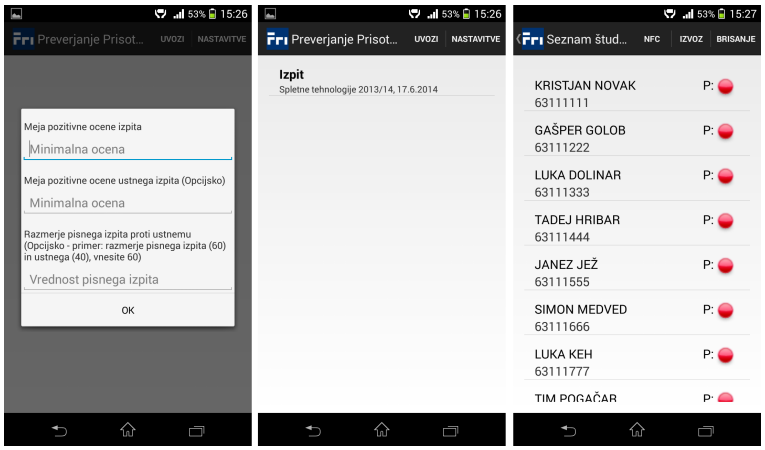
V primeru, da uporabnik uvaža seznam tipa *izpit*, se odpre dialog, ki od uporabnika zahteva, da vpiše (slika 4.6):

- mejo za pozitivno oceno pisnega izpita,
- mejo za pozitivno oceno ustnega izpita in
- vrednost pisnega izpita proti ustnemu.

V zadnje polje je potrebno vpisati število od 1 do 100, glede na to, koliko je vreden pisni izpit. Če končno oceno sestavlja 60% pisnega izpita in 40% ustnega izpita,



Slika 4.5: Dialog za izbiro imena in tipa seznama (v primeru uvoza iz sistema STUDIS, tipa seznama ni mogoče izbrati, saj smo ga že v prejšnjem koraku).



Slika 4.6: Dialog za vnos podrobnosti o seznamu tipa *izpit* (levo), seznam po uvozu (na sredini) in prikaz vseh študentov s seznama (desno).

je potrebno vnesti 60. Če ustni izpit ni potreben, lahko uporabnik vnese vrednost 0, v primeru, da ne vnese nobene vrednosti, pa sta izpita enakovredna.

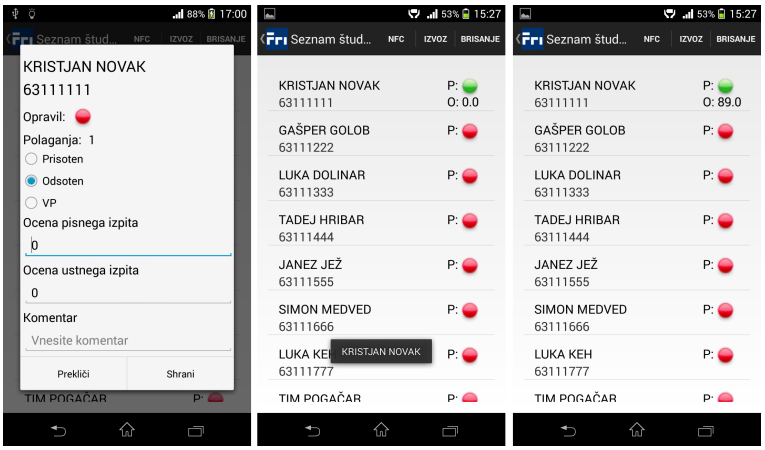
Ob končnem uvozu seznama se ta pojavi na zaslonu in ga je možno izbrati (slika 4.6). S to akcijo se odpre nova aktivnost, ki prikaže vse študente iz izbranega seznama. Vsak študent je prikazan v svoji vrstici, ki jo sestavljajo naslednji elementi (slika 4.6):

- ime in priimek študenta,
- vpisna številka študenta,
- indikator prisotnosti in
- končna ocena.

.

Indikator prisotnosti ima tri stanja. Rdeča barva predstavlja študentovo odsotnost, zelena predstavlja prisotnost, rumena pa vrnjeno prijavo študentu.

Profesor lahko vsakega študenta izbere in ureja njegove podrobnosti (slika 4.7). Vsakemu študentu je mogoče vpisati doseženo pisno in ustno oceno, ter dodati komentar. Omogočeno pa je tudi ročno spreminjanje študentove prisotnosti oziroma

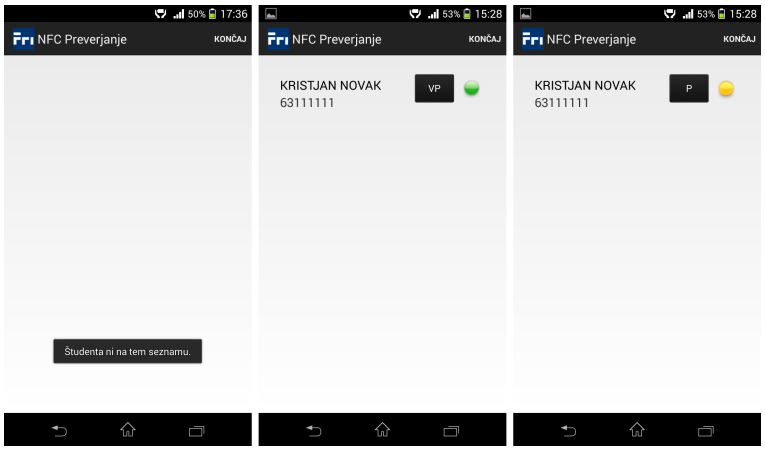


Slika 4.7: Dialog s podrobnostmi o študentu (levo), izgled zaslona ob preverjanju prisotnosti z izkaznicami (na sredini) in izgled študenta, ki je prisoten in ima vpisane vse ocene (desno).

odsotnosti. Ta opcija bi prišla v poštev, če bi študent pozabil svojo izkaznico in bi bilo potrebno njegovo prisotnost zabeležiti ročno.

Za preverjanje prisotnosti študentov z NFC izkaznicami, je potrebno imeti odprt željen seznam in približati izkaznico, ki jo želimo odčitati. Na zaslonu se izpiše obvestilo (ang. *toast*) s študentovim imenom in priimkom, hkrati pa se tudi spremeni stanje študenta na *prisoten*, kar je razvidno tudi po indikatorju prisotnosti (slika 4.7).

Prisotnost z izkaznicami je mogoče preverjati tudi tako, da v meniju uporabnik izbere opcijo *NFC*. Ta akcija bo uporabniku odprla novo prazno aktivnost, v primeru, da ni še nobenega študenta označil kot prisotnega. V nasprotnem primeru bodo že prisotni študentje prikazani na zaslonu. Za preverjanje prisotnosti je tudi v tem primeru potrebno približati izkaznico študenta. Razlika med tem in malo prej omenjenim preverjanjem je, da se tu študentje dodajajo na zaslon in imajo dodan gumb *VP* za hitro vračanje prijave (slika 4.8). V primeru, da aplikacija prebrano izkaznico ni uspela povezati z nobenim študentom iz tega seznama, to uporabniku sporoči (slika 4.8). To preverjanje prisotnosti lahko zaključimo z opcijo v meniju *končaj* (slika 4.8), ali pa s pritiskom na gumb *nazaj* (ang. *back*).



Slika 4.8: Sporočilo, da študenta, katerega izkaznico je uporabnik prebral ni na tem seznamu (levo). Izgled zaslona ob preverjanju prisotnosti z izkaznico in ob uspešni povezavi katrice s študentom (na sredini) in izgled zaslona, ob vrnitvi prijave študentu (desno).

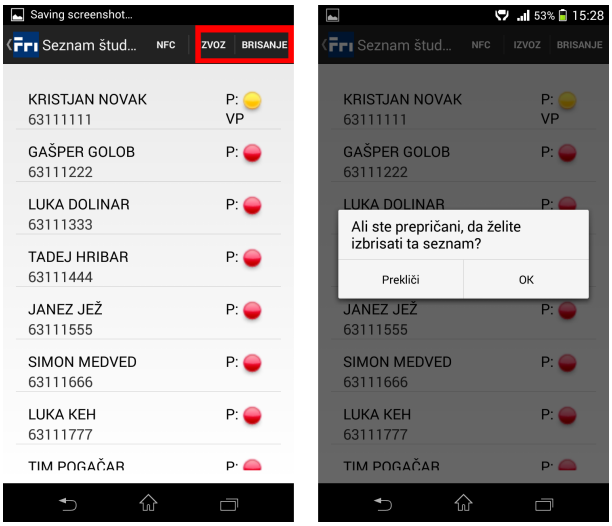
V aktivnosti, ki prikazuje vse študente s seznama, je mogoče seznam tudi izvoziti in ga izbrisati (slika 4.9). V primeru, da želi uporabnik seznam izvoziti, se bo ta shranil v obliki `.csv` datoteke. Izvoženo datoteko lahko uporabnik najde v direktoriju *PreverjanjePrisotnosti-Izvozi*. Če uporabnik želi izbrisati seznam, ga aplikacija najprej opozori na brisanje seznama, nato pa ga po potrditvi izbriše (slika 4.9).

Vaje

Če uporabnik uvaža seznam tipa *vaje*, se odpre dialog z naslednjimi polji (slika 4.10):

- minimalna prisotnost in
- minimalna povprečna ocena za uspešno opravljene vaje.

Ko je seznam uvožen, ga je možno izbrati (slika 4.10). Ta akcija bo začasno zaustavila trenutno odprto aktivnost in odprla novo, ki prikazuje vse študente z izbranega seznama. Za vsakega študenta so prikazani naslednji podatki (slika 4.10):



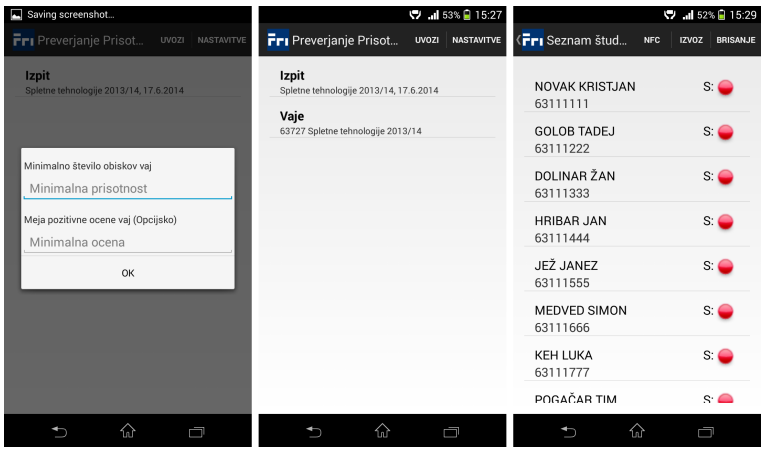
Slika 4.9: Opciji za izvoz in brisanje seznama (levo) ter potrditveno opozorilo o brisanju seznama (desno).

- ime in priimek študenta,
- vpisna številka študenta,
- indikator prisotnosti na zadnjih vajah in
- zadnja vpisana ocena.

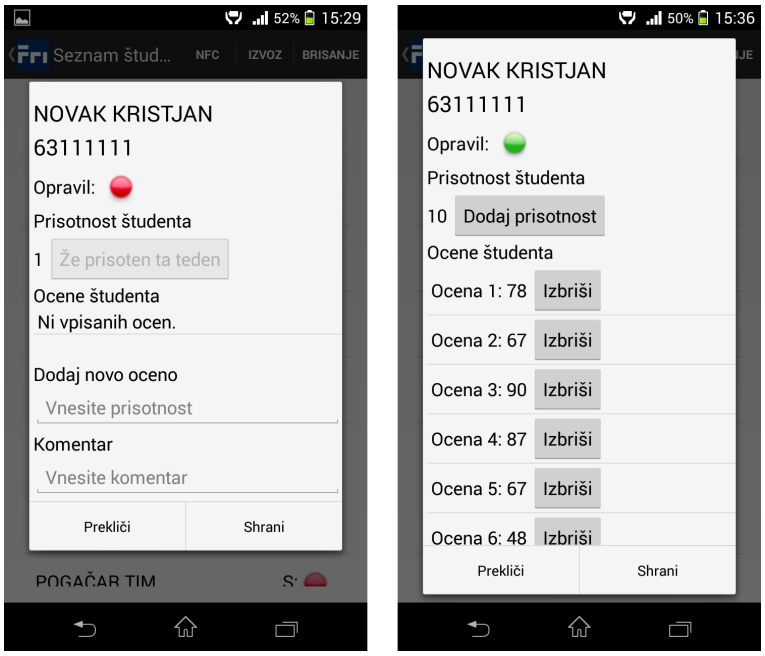
Indikator prisotnosti ima tri stanja. Rdeča pomeni, da študenta na vajah v tekočem tednu ni bilo. Zelena pomeni, da je bil študent prisoten in ima vpisano oceno tekočega tedna. Rumena pa pomeni, da je bil študent prisoten, vendar nima vpisane ocene. Indikator prisotnosti se ponastavi vsak ponedeljek ob 0:00, kar pomeni, da teden vaj traja od ponedeljka od 0:00 do nedelje do 23:59.

Profesor lahko vsakega študenta na seznamu izbere. Odprl se bo dialog s podrobnostmi o študentu in možnostjo ročnega dodajanja prisotnosti, ocen in komentarja (slika 4.11). Študentu je mogoče dodati poljubno število ocen, ki se prikazujejo v obliki seznama (slika 4.11). Vsako oceno s tega seznama ima uporabnik možnost tudi izbrisati s pritiskom na gumb *Izbriši*.

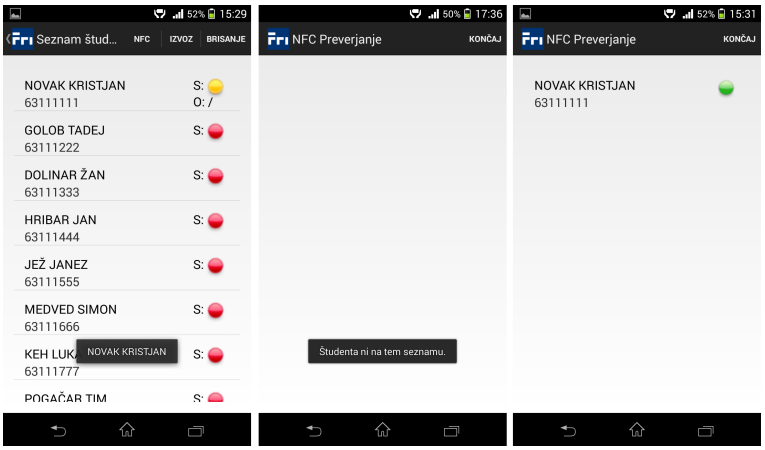
Za preverjanje prisotnosti študentov je potrebno imeti odprt seznam in približati izkaznico, ki jo uporabnik želi odčitati. Tako kot pri seznamu tipa *izpit*,



Slika 4.10: Dialog za vnos podrobnosti o seznamu (levo), zaslon po končanem uvozu (na sredini) ter prikaz vseh študentov uvoženega seznama (desno).



Slika 4.11: Dialog s podrobnostmi o študentu (levo) in izgled dialoga po nekaj tednih izvajanja vaj (desno).

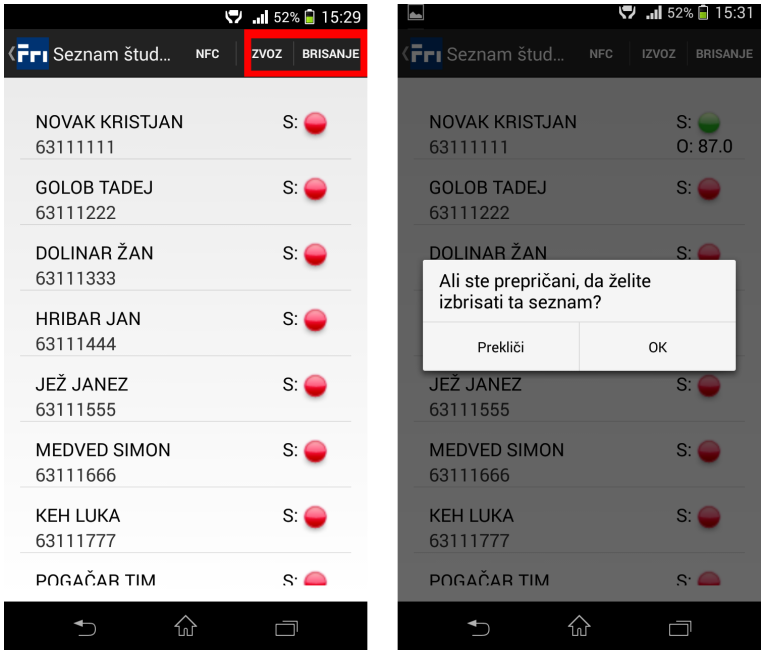


Slika 4.12: NFC preverjanje prisotnosti z izkaznicami ob odprtem seznamu (levo), obvestilo o neuspešnem povezovanju izkaznice s študentom (na sredini) ter zaslon ob uspešno povezanem študentu z izkaznico (desno).

se tudi tu izpiše študentovo ime in se zabeleži njegova prisotnost. Ta akcija je vidna tudi na ekranu, saj se indikator prisotnosti študenta, katerega prisotnost uporabnik preverja, obarva rumeno (slika 4.12).

Tudi pri tem tipu seznama ima uporabnik na voljo v meniju opcijo *NFC*. Ta opcija odpre novo aktivnost za preverjanje prisotnosti. Na začetku je zaslon prazen. Ko približamo izkaznico študenta, ki je na tem seznamu, pa se pojavi na ekranu njegovo ime (slika 4.12). V primeru, da izkaznice ni mogoče povezati z nobenim od študentov na tem seznamu, se uporabniku izpiše obvestilo, da tega študenta ni na seznamu (slika 4.12). To preverjanje lahko končamo s pritiskom na gumb *nazaj* (ang. *back*), ali pa z opcijo *končaj* v meniju.

V aktivnosti, ki prikazuje vse študente tega seznama, je mogoče seznam tudi izvoziti in izbrisati. Opcije za ti dve akciji lahko uporabnik najde v meniju aplikacije (slika 4.13). Ob izvozu seznama, se ta shrani v direktorij *PreverjanjePrisotnosti-Izvozi*, v primeru brisanja seznama pa nas aplikacija najprej opozori na brisanje in nato ob potrditvi seznam izbriše (slika 4.13).



Slika 4.13: Opciji za brisanje in izvoz seznama (levo) ter opozorilo o brisanju seznama (desno).

Poglavje 5

Sklepne ugotovitve

Predstavljena je bila mobilna aplikacija, ki omogoča lažje pregledovanje prisotnosti študentov na izpitih in vajah. Poleg tega omogoča še beleženje ocen in komentarjev za vsakega študenta posebej. Aplikacija z uporabo NFC tehnologije odčitava študentske izkaznice in na ta način potrdi študentovo prisotnost.

Program bo lahko uporabljal vsak profesor, ki ima pametni telefon ali tablični računalnik, ki ga poganja operacijski sistem Android 4.0 ali novejši. Seveda je za popolno delovanje potrebno na napravi imeti tudi NFC, ni pa to nujno potrebno, saj aplikacija deluje tudi brez te tehnologije. S tem bodo imeli profesorji olajšano preverjanje prisotnosti študentov, saj jim ne bo potrebno imeti listov s seznami študentov, ki so se prijavili na izpitni rok oziroma obiskujejo njegove vaje.

Aplikacija bo uporabna tudi v primerih, ko bo profesor pozabil seznam prijavljenih študentov, saj bo lahko s samo nekaj kliki seznam prijavljenih naložil na svojo napravo preko STUDIS sistema.

Aplikacija za prikaz na zaslonu uporablja privzete Android elemente, ki niso ravno najlepši. Tu je še veliko prostora za izboljšanje aplikacije in uporabniške izkušnje. Kot dodatno funkcionalnost bi lahko aplikacija imela še ročno povezovanje študenta z njegovo izkaznico. Ta funkcionalnost bi bila uporabna v primeru, da študent izgubi svojo prvotno izkaznico in nato dobi novo. V tem primeru bi bilo potrebno v seznamu popraviti identifikator izkaznice. To bi bilo mogoče urediti kar v aplikaciji.

Razvoj aplikacije je bil zasnovan kot prototipna rešitev, ki se bo v študijskem letu 2014/2015 uporabljala za testiranje delovanja in pridobitev informacij o upo-

rabniški izkušnji profesorja in tudi študentov.

Literatura

[1] Tom Igoe, Don Coleman, Brian Jepson, "Near Fieald Communication with Arduion, Android and PhoneGap", *Beginning NFC*, 2013.

[2] Rob Triggs (2014) What is NFC & how does it work? Dostopno na: http://www.androidauthority.com/what-is-nfc-270730/

[3] (2014) NFC Data Exchange Format (NDEF). Dostopno na: http://ibadrinath.blogspot.com/2012/07/nfc-data-exchange-format-ndef.html

[4] (2014) Near Field Communication. Dostopno na: http://developer.android.com/guide/topics/connectivity/nfc/index.html

[5] (2014) NFC Tag Types. Dostopno na: http://www.nfc.cc/technology/nfc-tag-types/

[6] (2014) The Tag Dispatch System. Dostopno na: http://developer.android.com/guide/topics/connectivity/nfc/nfc.html#tag-dispatch

[7] (2014) Using the Foreground Dispatch System. Dostopno na: http://developer.android.com/guide/topics/connectivity/nfc/advanced-nfc.html#foreground-dispatch

[8] (2014) MF3ICD81, MF3ICD41, MF3ICD21. Dostopno na: http://www.acs.com.hk/download-manual/2266/TDS_DESF_EV1.pdf

[9] (2014) Eclipse. Dostopno na: http://www.eclipse.org/org/

[10] (2014) Android SDK. Dostopno na:
http://www.webopedia.com/TERM/A/Android_SDK.html

[11] (2014) SQLite. Dostopno na:
http://www.sqlite.org/about.html

[12] (2014) Android Emulator. Dostopno na:
http://developer.android.com/tools/help/emulator.html

[13] (2014) Vertabelo. Dostopno na:
http://www.vertabelo.com/

[14] (2014) Sony Xperia SP. Dostopno na:
http://www.phonearena.com/phones/Sony-Xperia-SP_id7785

[15] (2014) Samsung Galaxy Tab 2. Dostopno na:
http://www.phonearena.com/phones/Samsung-Galaxy-Tab-2-10.1_id6962

[16] (2014) Implement Master/Detail Flows Across Handsets and Tablets. Dostopno na:
http://developer.android.com/training/implementing-navigation/descendant.html

[17] (2014) What is SHA Encryption? Dostopno na:
http://www.secpoint.com/what-is-sha-encryption.html

[18] (2014) Google Nexus S. Dostopno na:
http://www.phonearena.com/phones/Google-Nexus-S_id4990

[19] (2014) Google Nexus 10. Dostopno na:
http://www.phonearena.com/phones/Google-Nexus-10_id7551